Your first application

Keywords:

"how ASP.NET Core applications work"

"set up a development environment"

"creating your first web app"

"customizing and building your own applications"

"components that make up an ASP.NET Core application"

"general application-building process"

"a template provides the basic code required to build an application"

"Visual Studio templates"

"Visual Studio 2019 and ASP.NET Core 5.0 with .NET 5.0"

"working with the .NET CLI"

"GitHub repository for the book at https://github.com/andrewlock/asp-dot-net-core-in-action-2e"

"restore all the necessary dependencies, compile your application, and run it"

"a fully configured ASP.NET Core application"

"configure the web server, the middleware pipeline, and HTML generation"

"Program.cs and Startup.cs files"

"define the middleware pipeline for your application"

"how the app generates HTML in response to a request"

"components that make up the Razor Pages endpoint"

"basic understanding of how ASP.NET Core applications are put together"

"how ASP.NET Core applications handle requests"

Questions:

Q: What should you have after reading chapter 1?

A: A general idea of how ASP.NET Core applications work and when you should use them.

Q: What is the focus of the current chapter?

A: Creating your first web app.

Q: What will readers begin to understand as they work through the chapter?

A: The various components that make up an ASP.NET Core application.

Q: What is a template defined as?

A: A template provides the basic code required to build an application.

Q: What tool does the author use to create a basic ASP.NET Core application?

A: Visual Studio templates.

Q: What development tools are supported besides Visual Studio?

A: The .NET CLI.

Q: Where can the application code for the chapter be viewed?

A: In the GitHub repository for the book at https://github.com/andrewlock/asp-dot-net-core-in-action-2e.

Q: What are the initial steps after creating your application?

A: Restore all the necessary dependencies, compile your application, and run it.

Q: How many pages will the simple application have?

A: It will only have two different pages.

Q: What parts of an ASP.NET Core application will be introduced after running the app?

A: The web server, the middleware pipeline, and HTML generation.

Q: Which files contain virtually the entire configuration of your application?

A: Program.cs and Startup.cs files.

Q: What will you learn to define in your application configuration?

A: The middleware pipeline for your application.

Q: What will you see about Razor Pages?

A: The components that make up the Razor Pages endpoint.

Q: What is controlled by the Razor Pages endpoint?

A: What code is run in response to a request and how to define the HTML that should be returned.

Q: What should you have by the end of the chapter?

A: A basic understanding of how ASP.NET Core applications are put together.

Q: What will be reviewed before beginning the project?

A: How ASP.NET Core applications handle requests.

A brief overview of an ASP.NET Core application

Keywords:

"how a browser makes an HTTP request to a server and receives a response"

"dynamically generate that HTML depending on the particulars of the request"

"display different data depending on the current logged-in user"

"create a web app to display information about your company"

"the middleware pipeline and the endpoint middleware"

"the default cross-platform Kestrel server"

"generate an HttpContext object"

"HTTP.sys, only runs on Windows and can't be used with IIS"

"series of components that process the incoming request"

"endpoint middleware is responsible for calling the code that generates the final response"

"Razor Pages are responsible for generating the HTML"

"calling out to various services in response to the data contained in the original request"

"MVC or Razor Pages block"

"how to choose between them"

"generating HTML in chapters 7 and 8"

"Most ASP.NET Core applications follow this basic architecture"

"how the code corresponds to the outline in figure 2.1"

Questions:

Q: What does a browser do with an HTTP request and response?

A: A browser makes an HTTP request to a server and receives a response, which it uses to render HTML on the page.

Q: What allows you to dynamically generate HTML based on a request?

A: ASP.NET Core allows you to dynamically generate that HTML depending on the particulars of the request.

Q: What is an example of content that can change depending on the request?

A: Display different data depending on the current logged-in user.

Q: What kind of app could you create to display company information?

A: A simple ASP.NET Core app.

Q: What has been expanded in the ASP.NET Core application diagram in figure 2.1?

A: The middleware pipeline and the endpoint middleware.

Q: What is the default cross-platform ASP.NET Core web server?

A: The default cross-platform Kestrel server.

Q: What does Kestrel use to generate an HttpContext object?

A: The raw incoming network request.

Q: What is the main alternative to Kestrel?

A: HTTP.sys, only runs on Windows and can't be used with IIS.

Q: What does Kestrel hand over to the middleware pipeline?

A: The HttpContext.

Q: What is found in every ASP.NET Core application that processes incoming requests?

A: The middleware pipeline.

Q: What are some operations performed by middleware components?

A: Logging, handling exceptions, or serving static files.

Q: What is responsible for calling the code that generates the final response?

A: The endpoint middleware.

Q: What typically generates the HTML in an ASP.NET Core web app?

A: Razor Pages.

Q: Where is most of the business logic of an app typically found?

A: Razor Pages.

Q: What chapter covers Razor Pages and MVC controllers?

A: Chapter 4.

Q: What chapters cover generating HTML?

A: Chapters 7 and 8.

Q: What kind of architecture do most ASP.NET Core applications follow?

A: Most ASP.NET Core applications follow this basic architecture.

Q: What does the example in this chapter demonstrate?

A: How the code corresponds to the outline in figure 2.1.

Creating your first ASP.NET Core application

Keywords:

"start building applications with ASP.NET Core in many different ways"

"tools and operating system you're using"

"Visual Studio 2019 template"

"templates from the .NET CLI or Visual Studio for Mac"

"Visual Studio 2019 and ASP.NET Core 5.0 with .NET 5.0"

"four basic steps"

"Generate—Create the base application from a template"

"Restore—Restore all the packages and dependencies"

"Build—Compile the application"

"Run—Run the compiled application"

"many ASP.NET Core templates for building different types of applications"

"Razor Pages web application"

"MVC (Model-View-Controller) application"

"Web API application"

"return data in a format that can be consumed by single-page applications (SPAs) and APIs"

"focus on the Razor Pages template"

Questions:

Q: What does the way you start building ASP.NET Core applications depend on?

A: The tools and operating system you're using.

Q: What template is the example in this chapter based on?

A: Visual Studio 2019 template.

Q: Which tools' templates can you use to follow along with this chapter?

A: Templates from the .NET CLI or Visual Studio for Mac.

Q: What version of ASP.NET Core and .NET does this chapter use?

A: ASP.NET Core 5.0 with .NET 5.0.

Q: What is the first step in getting an application up and running?

A: Generate—Create the base application from a template.

Q: What is the purpose of the Restore step?

A: Restore all the packages and dependencies to the local project folder using NuGet.

Q: What happens during the Build step?

A: Compile the application and generate all the necessary assets.

Q: What is the final step in running the application?

A: Run—Run the compiled application.

Q: What kinds of templates do Visual Studio and the .NET CLI include?

A: Many ASP.NET Core templates for building different types of applications.

Q: What do Razor Pages applications generate?

A: Razor Pages applications generate HTML on the server.

Q: What kind of applications use traditional MVC controllers?

A: MVC (Model-View-Controller) applications.

Q: What format do Web API applications return data in?

A: A format that can be consumed by single-page applications (SPAs) and APIs.

Q: What client-side applications are Web API apps typically used with?

A: Angular and React.js or mobile applications.

Q: Which template is the focus of this chapter?

A: The Razor Pages template.

Using a template to get started

Keywords:

"automatically configuring many of the fundamental pieces"

"standard templates for building web applications, console applications, and class libraries"

"a project is a unit of deployment"

"compiled into a .dll file or an executable"

"each separate app is a separate project"

"multiple projects can be built and developed at once in a solution"

"Choose Create a New Project from the splash screen"

"choose ASP.NET Core Web Application"

"select the C# language template"

"use WebApplication1 as both the project and solution name"

"Ensure .NET Core is selected"

"Select ASP.NET Core 5.0"

"Select ASP.NET Core Web App to create a Razor Pages web application"

"Ensure No Authentication is specified"

"Ensure Configure for HTTPS is checked"

"Ensure Enable Docker Support is unchecked"

"Visual Studio has created and added a number of files to your project"

"create a similar template using the .NET CLI"

"Open a PowerShell or cmd prompt"

"you now have the basic files required to build and run your first ASP.NET Core application"

Questions:

Q: What does using a template help with when starting an application?

A: Automatically configuring many of the fundamental pieces.

Q: What types of applications can be built using standard templates in Visual Studio and the .NET CLI?

A: Web applications, console applications, and class libraries.

Q: What is a project in .NET considered to be?

A: A unit of deployment.

Q: What is each separate app in .NET considered to be?

A: A separate project.

Q: What allows multiple projects to be developed at once?

A: A solution.

Q: What is the first step to create a web application in Visual Studio?

A: Choose Create a New Project from the splash screen.

Q: Which template should you select when creating an ASP.NET Core web application?

A: ASP.NET Core Web Application.

Q: What language template must be selected for the project?

A: The C# language template.

Q: What is an example name for both the project and solution?

A: WebApplication1.

Q: What must be selected to proceed with ASP.NET Core Web App creation?

A: ASP.NET Core Web App.

Q: What should you ensure is specified under authentication settings?

A: No Authentication.

Q: What must be checked to configure security during project setup?

A: Configure HTTPS.

Q: What should be unchecked regarding containerization support?

A: Enable Docker Support.

Q: What happens after Visual Studio finishes generating the application?

A: You'll be presented with an introductory page about ASP.NET Core.

Q: How can a similar template be created without Visual Studio?

A: Using the .NET CLI.

Q: Where should you run the .NET CLI commands to create a project?

A: In a PowerShell or cmd prompt on Windows or a terminal session on Linux or macOS.

Q: What do you have after completing the setup in either Visual Studio or .NET CLI?

A: The basic files required to build and run your first ASP.NET Core application.

Building the application

Keywords:

"ensure all the dependencies used by your project are copied to your local directory"

"compile your application so that it can be run"

"automatically restore packages when they first create your project"

"manually restore packages using dotnet restore"

"Build > Build Solution"

"Ctrl-Shift-B"

"dotnet build from the command line"

"output window shows the progress of the build"

"dotnet build console commands from the Package Manager Console"

"automatically build your application when you run it if they detect that a file has changed"

Questions:

Q: What are the two remaining steps before running your application?

A: Ensure all the dependencies used by your project are copied to your local directory, and compile your application so that it can be run.

Q: When are packages automatically restored in the .NET CLI or Visual Studio?

A: When they first create your project.

Q: What command was required in earlier versions of the .NET CLI before 2.0?

A: Manually restore packages using dotnet restore.

Q: What is one way to compile your application in Visual Studio?

A: Choose Build > Build Solution.

Q: What keyboard shortcut can be used to compile the application in Visual Studio?

A: Ctrl-Shift-B.

Q: What command can be used to build the application from the command line?

A: dotnet build.

Q: Where can you run dotnet build console commands within Visual Studio?

A: From the Package Manager Console.

Q: What does the output window show during the build process in Visual Studio?

A: The progress of the build.

Q: When do Visual Studio and .NET CLI tools automatically build your application?

A: When you run it if they detect that a file has changed.

NuGet packages and the .NET command-line interface

Keywords:

".NET command-line interface (CLI)"

"dotnet restore"

"dotnet build"

"dotnet run"

"run inside your project folder"

"dependencies on various external libraries"

"NuGet package manager"

"dependencies are listed in the project"

"local copies of each dependency"

".csproj file"

"PackageReference node"

"restore process typically happens implicitly"

"compile your application using dotnet build"

"dotnet --help"

"dotnet new --help"

Questions:

Q: What is the purpose of the .NET command-line interface (CLI)?

A: It provides basic commands for creating, building, and running .NET 5.0 applications.

Q: Name the three most common .NET CLI commands used during development.

A: dotnet restore, dotnet build, and dotnet run.

Q: Where should these dotnet commands be run?

A: Inside your project folder.

Q: How are dependencies managed in most ASP.NET Core applications?

A: Through the NuGet package manager.

Q: What does the dotnet restore command do?

A: It ensures your application's NuGet dependencies are copied to your project folder.

Q: Where are ASP.NET Core project dependencies listed?

A: In the project's .csproj file.

Q: What does the .csproj file contain to list dependencies?

A: PackageReference nodes.

Q: When does the restore process typically happen?

A: Implicitly when you build or run your application.

Q: What does the dotnet build command do?

A: It compiles your application and checks for errors.

Q: Which command runs the compiled application?

A: dotnet run.

Q: How can you see the full list of available dotnet commands?

A: By running dotnet --help.

Q: How can you see options available for a specific command like 'new'?

A: By running dotnet new --help.

Running the web application

Keywords:

"click the green arrow on the toolbar next to IIS Express"

"press the F5 shortcut"

"Visual Studio will automatically open a web browser window"

"run the application from the command line with the .NET CLI tools using dotnet run"

"install the development certificate"

"browser doesn’t display warnings about an invalid certificate"

"simple Welcome banner and a link to the official Microsoft documentation for ASP.NET Core"

"Home and Privacy"

"header containing the links and the application title, 'WebApplication1'"

"title of the page, as shown in the tab of the browser, changes to match the current page"

"can only view the application on the same computer that is running it"

"application isn’t exposed to the internet yet"

"learn how to publish and deploy your application in chapter 16"

Questions:

Q: How can you run your first application in Visual Studio?

A: You can click the green arrow on the toolbar next to IIS Express or press the F5 shortcut.

Q: What happens the first time you run the application from Visual Studio?

A: You will be prompted to install the development certificate.

Q: What does installing the development certificate ensure?

A: It ensures your browser doesn’t display warnings about an invalid certificate.

Q: What does the default page of the application show?

A: A simple Welcome banner and a link to the official Microsoft documentation for ASP.NET Core.

Q: What are the two links at the top of the default page?

A: Home and Privacy.

Q: What is consistent on both the Home and Privacy pages?

A: The header containing the links and the application title, "WebApplication1".

Q: What changes when you navigate between pages in the browser?

A: The title of the page, as shown in the tab of the browser, changes to match the current page.

Q: Where can you view the running application?

A: Only on the same computer that is running it at the moment.

Q: Is the application exposed to the internet by default?

A: No, the application isn’t exposed to the internet yet.

Q: In which chapter will you learn about publishing and deploying your application?

A: Chapter 16.

Understanding the project layout

Keywords:

"creating an application from a template"

"the basic web application template doesn't contain a huge number of files and folders"

"the main project, WebApplication1, is nested in a top-level directory with the name of the solution"

"solution (.sln) file for use by Visual Studio"

"Visual Studio uses the concept of a solution to work with multiple projects"

"you won't have a .sln or Git files unless you generate them explicitly using additional .NET CLI templates"

"three subfolders—Pages, Properties, and wwwroot"

"Pages contains the Razor Pages files"

"Properties folder contains a single file, launchSettings.json"

"wwwroot folder is special, in that it's the only folder in your application that browsers are allowed to directly access"

"store your CSS, JavaScript, images, or static HTML files in here"

"Dependencies and Connected Services, but they don't have corresponding folders on disk"

"two JSON files: appsettings.json and appsettings.Development.json"

"WebApplication1.csproj, as it describes how to build your project"

"Visual Studio shows two C# files in the project folder—Program.cs and Startup.cs"

"fundamental classes are responsible for configuring and running your application"

Questions:

Q: What does the basic web application template contain in terms of files and folders?

A: It doesn't contain a huge number of files and folders.

Q: Where is the main project WebApplication1 nested?

A: It is nested in a top-level directory with the name of the solution, which is also WebApplication1.

Q: What file is used by Visual Studio to work with multiple projects?

A: The solution (.sln) file.

Q: What subfolders does the project folder contain?

A: Pages, Properties, and wwwroot.

Q: What does the Pages folder contain?

A: The Razor Pages files you'll use to build your application.

Q: What file does the Properties folder contain and what does it control?

A: launchSettings.json, which controls how Visual Studio will run and debug the application.

Q: Why is the wwwroot folder special?

A: It's the only folder in your application that browsers are allowed to directly access.

Q: What types of files can you store in the wwwroot folder?

A: CSS, JavaScript, images, or static HTML files.

Q: What do the Dependencies and Connected Services nodes show?

A: A collection of all the dependencies, such as NuGet packages, and remote services that the project relies on.

Q: What are the names of the two JSON files in the root of your project folder?

A: appsettings.json and appsettings.Development.json.

Q: What is the significance of WebApplication1.csproj?

A: It describes how to build your project.

Q: Which two C# files does Visual Studio show in the project folder?

A: Program.cs and Startup.cs.

Q: What are Program.cs and Startup.cs responsible for?

A: Configuring and running your application.

The .csproj project file: Defining your dependencies

Keywords:

".csproj file is the project file for .NET applications"

"defines the type of project being built"

"which platform the project targets"

"which NuGet packages the project depends on"

"No GUIDs"

"Implicit file includes"

"No paths to NuGet package .dll files"

"files are automatically compiled"

"reference the NuGet package directly in your .csproj"

"project file far more compact than you'll be used to"

"Sdk attribute on the Project element includes default settings"

"TargetFramework element describes the framework your application will run on"

"netcoreapp3.1 value"

"net5.0"

"add additional NuGet packages using the PackageReference element"

"app doesn't reference any NuGet packages at all"

"simplified project file format is much easier to edit by hand"

"Visual Studio will update the project file itself"

"project file defines everything Visual Studio and the .NET CLI need to build your app"

Questions:

Q: What does the .csproj file define for a .NET application?

A: It defines the type of project being built, which platform the project targets, and which NuGet packages the project depends on.

Q: What change has been made regarding GUIDs in the new .csproj file format?

A: GUIDs are rarely used in the project file now.

Q: How are files included in the build process in the new .csproj format?

A: Files are automatically compiled without needing to be listed explicitly.

Q: How does the new .csproj format handle NuGet package references?

A: You can reference the NuGet package directly in your .csproj without specifying the path to .dll files.

Q: What attribute in the project file includes default settings on how to build your project?

A: The Sdk attribute on the Project element.

Q: What does the TargetFramework element describe?

A: The framework your application will run on.

Q: What value does the TargetFramework have for .NET Core 3.1 projects?

A: netcoreapp 3.1

Q: What value does the TargetFramework have for .NET 5.0 projects?

A: net5.0

Q: How can you add additional NuGet packages to your project file?

A: By using the PackageReference element.

Q: How does Visual Studio handle project file changes when using its GUI?

A: Visual Studio will update the project file itself.

Q: What does the project file define for Visual Studio and the .NET CLI?

A: Everything they need to build your app.

The Program class: Building a web host

Keywords:

"All ASP.NET Core applications start in the same way as .NET Console applications—with a Program.cs file"

"static void Main function"

"Main method must exist and is called whenever you start your web application"

".NET 5.0 and C# 9 introduced 'top-level statements'"

"Main entry point is used to build and run an IHost instance"

"IHost is the core of your ASP.NET Core application"

"IHost contains the application configuration and the Kestrel server"

"Main function contains all the basic initialization code required to create a web server and to start listening for requests"

"IHostBuilder created by the call to CreateDefaultBuilder"

"builder object to configure a complex object"

"Startup class referenced in the generic UseStartup<> method"

"Startup class is where you configure your app's services and define your middleware pipeline"

"Program is where you configure the infrastructure of your application"

"Startup is where you define which components and features your application uses"

"Program class is generally similar but Startup classes differ significantly"

"CreateDefaultBuilder is a static helper method"

"ConfigureWebHostDefaults uses a WebHostBuilder object to configure Kestrel"

"call to Build produces the IHost instance"

"call to Run starts the HTTP server listening"

"application is fully operational and can respond to its first request from a remote browser"

Questions:

Q: What file do all ASP.NET Core applications start with?

A: Program.cs file.

Q: What function must exist and is called whenever you start your web application?

A: The static void Main function.

Q: What is the Main entry point used for in ASP.NET Core applications?

A: To build and run an IHost instance.

Q: What does the IHost contain in an ASP.NET Core application?

A: The application configuration and the Kestrel server.

Q: What does the Main function do in the Program.cs file?

A: It contains all the basic initialization code required to create a web server and start listening for requests.

Q: How is the IHostBuilder created in the Main function?

A: By the call to CreateDefaultBuilder.

Q: What is the role of the Startup class in an ASP.NET Core application?

A: To configure your app's services and define your middleware pipeline.

Q: What is the difference between Program and Startup classes?

A: Program configures the infrastructure, while Startup defines components, features, and middleware.

Q: What helper method simplifies bootstrapping your app by creating an IHostBuilder?

A: CreateDefaultBuilder.

Q: What does the ConfigureWebHostDefaults method do?

A: It uses a WebHostBuilder object to configure Kestrel to listen for HTTP requests.

Q: What does the call to Build produce?

A: The IHost instance.

Q: What does the call to Run do?

A: It starts the HTTP server listening.

Q: When is the application fully operational?

A: When it can respond to its first request from a remote browser.

The Startup class: Configuring your application

Keywords:

"Program is responsible for configuring a lot of the infrastructure for your app"

"Startup class is responsible for configuring two main aspects of your application"

"Service registration"

"Middleware and endpoints"

"service registration in ConfigureServices"

"middleware configuration in Configure"

"IHostBuilder created in Program calls ConfigureServices and then Configure"

"services registered in the ConfigureServices method are available to the Configure method"

"an IHost is created by calling Build() on the IHostBuilder"

"Startup class doesn't implement an interface as such"

"methods are invoked by using reflection"

"predefined names of Configure and ConfigureServices"

"reflection in .NET allows you to obtain information about types in your application at runtime"

"reflection to create instances of classes at runtime and to invoke and access them"

"ConfigureServices and Configure follow on from each other"

"contribute to the application's configuration as a whole"

Questions:

Q: What two main aspects of your application does the Startup class configure?

A: Service registration and middleware and endpoints.

Q: In which Startup method do you configure service registration?

A: ConfigureServices.

Q: In which Startup method do you configure middleware?

A: Configure.

Q: What calls ConfigureServices and then Configure?

A: The IHostBuilder created in Program.

Q: Are services registered in ConfigureServices available to the Configure method?

A: Yes, they are available to the Configure method.

Q: How is the Startup class's methods invoked?

A: By using reflection to find methods with the predefined names Configure and ConfigureServices.

Q: Does the Startup class implement an interface?

A: No, it doesn't implement an interface as such.

Q: What does reflection in .NET allow you to do?

A: Obtain information about types in your application at runtime and create instances of classes and invoke them.

Q: How do ConfigureServices and Configure relate to each other?

A: ConfigureServices and Configure follow on from each other and contribute to the application's configuration as a whole.

Adding and configuring services

Keywords:

"ASP.NET Core uses small, modular components for each distinct feature"

"individual features to evolve separately"

"modular components are exposed as one or more services"

"service refers to any class that provides functionality to an application"

"TaxCalculator that calculates the tax due on a particular product"

"ShippingCostService that calculates the cost of shipping"

"OrderTotalCalculatorService might use both of these services"

"single responsibility principle"

"single responsibility principle (SRP) states that every class should be responsible for only a single piece of functionality"

"use the new keyword and create an instance of a service whenever you need it"

"tightly couples your code to the specific implementation"

"dependency injection or the Inversion of Control (IoC) principle"

"register the dependencies of your application into a 'container'"

"registration is performed in the ConfigureServices method"

"AddRazorPages() method is an extension method that encapsulates all the code required to set up the Razor Pages services"

"IServiceCollection is a list of every known service that your application will need to use"

"defining how your application responds to HTTP requests"

Questions:

Q: What design practice does ASP.NET Core use for each distinct feature?

A: ASP.NET Core uses small, modular components for each distinct feature.

Q: What does the term "service" refer to in ASP.NET Core?

A: Service refers to any class that provides functionality to an application.

Q: What principle states that every class should be responsible for only a single piece of functionality?

A: The single responsibility principle (SRP).

Q: What is a downside of using the new keyword to create an instance of a service whenever needed?

A: It tightly couples your code to the specific implementation.

Q: What technique allows a service to declare dependencies and have another class fill them?

A: Dependency injection or the Inversion of Control (IoC) principle.

Q: Where are dependencies registered in an ASP.NET Core application?

A: In the ConfigureServices method.

Q: What does the AddRazorPages() method do?

A: It is an extension method that encapsulates all the code required to set up the Razor Pages services.

Q: What is IServiceCollection?

A: IServiceCollection is a list of every known service that your application will need to use.

Q: What is the final configuration step after setting up services?

A: Defining how your application responds to HTTP requests.

Defining how requests are handled with middleware

Keywords:

"IHostBuilder and Startup classes"

"registered your services with the IoC container"

"Configure method for the template application"

"middleware pipeline for the application"

"middleware consists of small components that execute in sequence"

"IApplicationBuilder that's passed to the Configure method"

"order of the calls in this method is important"

"Middleware can only use objects created by previous middleware"

"IWebHostEnvironment parameter is used to provide different behavior"

"EnvironmentName is set to 'Development'"

"DeveloperExceptionPageMiddleware"

"ExceptionHandlerMiddleware is registered"

"Http Redirect Middleware"

"app.UseHttpsRedirection()"

"middleware is responsible for handling requests for static files"

"routing middleware and the endpoint middleware"

"MapRazorPages"

"AuthorizationMiddleware between the routing middleware and the endpoint middleware"

"StaticFileMiddleware"

"Razor Pages and how they generate HTML"

Questions:

Q: What does the Configure method in the Startup class define?

A: The Configure method defines the middleware pipeline for the application.

Q: Why is the order of middleware calls in the Configure method important?

A: Middleware can only use objects created by previous middleware in the pipeline.

Q: What does the IWebHostEnvironment parameter provide in the Configure method?

A: It provides different behavior when you're in a development environment.

Q: What middleware is added in development to provide detailed exception information?

A: DeveloperExceptionPageMiddleware.

Q: What middleware is registered in production to handle exceptions?

A: ExceptionHandlerMiddleware.

Q: What is the purpose of the Http Redirect Middleware?

A: It ensures your application only responds to secure (HTTPS) requests.

Q: What does the static file middleware do when a request for a static file arrives?

A: It returns the file if it exists; otherwise, it ignores the request so the next middleware can handle it.

Q: What two middleware components are responsible for interpreting requests and generating HTML?

A: The routing middleware and the endpoint middleware.

Q: What method do you call to specify that you wish to use Razor Page endpoints?

A: MapRazorPages.

Q: Where is the AuthorizationMiddleware added in the middleware pipeline?

A: Between the routing middleware and the endpoint middleware.

Q: What middleware serves image and CSS files to the user?

A: StaticFileMiddleware.

Generating responses with Razor Pages

Keywords:

"middleware pipeline"

"final piece of middleware in a pipeline is the endpoint middleware"

"routing middleware to match a request URL's path to a configured route"

"path is the remainder of the request URL, once the domain has been removed"

"routing middleware notes the selected Razor Page in the request's HttpContext"

"endpoint middleware executes the Razor Page to generate the HTML response"

"Razor Pages generate HTML using the Razor syntax"

"page handlers to add business logic and behavior to your Razor Pages"

Questions:

Q: What is normally the final piece of middleware in an ASP.NET Core middleware pipeline?

A: The final piece of middleware in a pipeline is the endpoint middleware.

Q: What does the routing middleware do with the request URL's path?

A: It matches the path to a configured route which defines which Razor Page to invoke.

Q: How is a "path" defined in the context of a request URL?

A: A path is the remainder of the request URL, once the domain has been removed.

Q: After selecting a Razor Page, what does the routing middleware do with it?

A: It notes the selected Razor Page in the request's HttpContext and continues executing the middleware pipeline.

Q: What is the role of the endpoint middleware in handling a request?

A: The endpoint middleware executes the Razor Page to generate the HTML response and sends it back to the browser.

Q: What will the next section discuss after this passage?

A: How Razor Pages generate HTML using the Razor syntax.

Q: What is the purpose of page handlers in Razor Pages?

A: To add business logic and behavior to your Razor Pages.

Generating HTML with Razor Pages

Keywords:

"Razor Pages are stored in .cshtml files"

"routing middleware maps request URL paths to a single Razor Page"

"the @page directive on the first line of the Razor Page"

"the @model directive"

"PageModel is called PrivacyModel"

"static HTML is always valid in a Razor Page and will be rendered 'as is'"

"you can write ordinary C# code in Razor templates by using this construct: @{ /* C# code here */ }"

"ViewData dictionary"

"you can dynamically write C# variables to the HTML stream using the @ symbol"

"Razor Pages have the concept of layouts"

"layouts are 'base' templates that define the common elements of your application"

"layouts in the Pages/Shared folder of your project"

"limit the code in your .cshtml file to presentational concerns only"

"complex logic, code to access services such as a database, and data manipulation should be handled in the PageModel"

Questions:

Q: Where are Razor Pages stored in an ASP.NET Core project?

A: Razor Pages are stored in .cshtml files within the Pages folder of your project.

Q: What does the @page directive on the first line of a Razor Page indicate?

A: It tells ASP.NET Core that the .cshtml file is a Razor Page.

Q: What is the purpose of the @model directive in a Razor Page?

A: It defines which PageModel in your project the Razor Page is associated with.

Q: How is static HTML treated inside a Razor Page?

A: Static HTML is always valid and will be rendered "as is" in the response.

Q: How can you include C# code in Razor templates?

A: By using the construct @{ /* C# code here */ }.

Q: What is the role of the ViewData dictionary in Razor Pages?

A: It stores values like the title of the page, which can be written to the response dynamically.

Q: How do Razor Pages combine dynamic C# variables with HTML?

A: By using the @ symbol to write C# variables to the HTML stream.

Q: What are layouts in Razor Pages?

A: Layouts are "base" templates that define common elements like headers and footers for your application.

Q: Where can layouts typically be found in the project?

A: In the Pages/Shared folder of your project.

Q: What kind of code should be limited in the .cshtml files?

A: You should limit code to presentational concerns only.

Q: Where should complex logic and data manipulation be handled?

A: In the PageModel instead of the .cshtml file.

Handling request logic with PageModels and handlers

Keywords:

"the @page directive in a .cshtml file marks the page as a Razor Page"

"placed in a file commonly known as a 'code behind' file that has a .cs extension"

"Page models should derive from the PageModel base class"

"page handler is a method that runs in response to a request"

"Page handlers are driven by convention"

"Page models can use dependency injection to interact with other services"

"named by convention, based on the HTTP verb that they respond to"

"return either void, indicating that the Razor Page's template should be rendered, or an IActionResult"

"Every HTTP request includes a verb that indicates the 'type' of the request"

"the default verb is GET"

"the second most common verb is POST"

"Dependency injection is used to inject an ILogger<PrivacyModel> instance"

"you can access additional services in your page model by accepting them as parameters in the constructor"

"framework for performing arbitrarily complex functions in response to a request"

"separated the execution of these methods from the generation of the HTML itself"

"Privacy.cshtml Razor Page executes the OnGet handler and generates the HTML response"

"request passes through the middleware pipeline before being handled by the endpoint middleware"

"overview of how an entire application is configured and how it handles a request using Razor Pages"

Questions:

Q: What does the @page directive in a .cshtml file do?

A: It marks the page as a Razor Page.

Q: Where is the page model for a Razor Page typically placed?

A: In a file commonly known as a "code behind" file that has a .cs extension.

Q: From which base class should page models be derived?

A: From the PageModel base class.

Q: What is a page handler?

A: A page handler is a method that runs in response to a request.

Q: What are page handlers typically named by?

A: By convention, based on the HTTP verb that they respond to.

Q: What are the typical return types for a page handler method?

A: They return either void or an IActionResult.

Q: What does the GET verb in an HTTP request do?

A: It fetches a resource from the server so you can view it.

Q: What is the second most common HTTP verb?

A: POST.

Q: What is injected into the PrivacyModel using dependency injection?

A: An ILogger<PrivacyModel> instance.

Q: How can additional services be accessed in a page model?

A: By accepting them as parameters in the constructor.

Q: What is the key advantage of having a page model even if it only renders HTML?

A: You now have a framework for performing arbitrarily complex functions in response to a request.

Q: Why is separating method execution from HTML generation beneficial?

A: You're less likely to introduce bugs when updating logic or UI.

Q: Which handler is executed by the Privacy.cshtml Razor Page?

A: The OnGet handler.

Q: What does the Privacy.cshtml Razor Page generate after executing the handler?

A: The HTML response.

Q: Where does the request pass before being handled by the endpoint middleware?

A: Through the middleware pipeline.

Q: What does the passage provide an overview of?

A: How an entire application is configured and how it handles a request using Razor Pages.